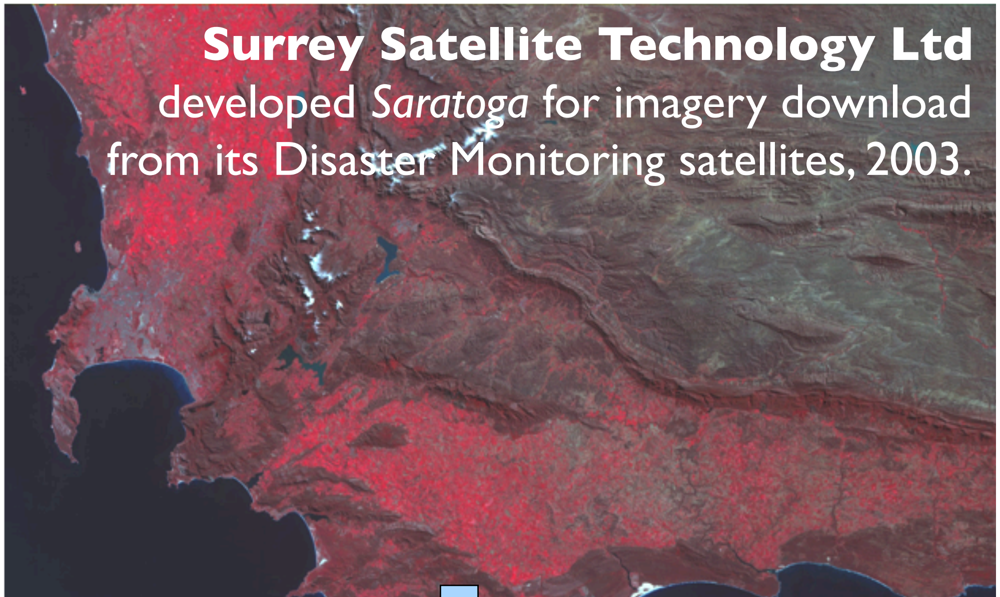
Saratoga: scalable, speedy data delivery for sensor networks

Private sensor networks

- Must deliver sensor data – very quickly.
- Want to use Internet technologies – cheap, reliable, robust.
- Want more speed than TCP can offer.
- Congestion is not a problem; private single-owner managed network with scheduled traffic, single flow per link with no competition. This is not the shared public Internet!
- Sensor capabilities are ever-increasing (side-effects of Moore's law). Need to scale for ever-growing data sizes.
- Support for streaming and simultaneous delivery to multiple receivers is also useful.
- ***Saratoga* protocol designed to meet these needs.**

Saratoga's development



Saratoga redesigned, specified to the Internet Engineering Task Force, 2007.

NASA Glenn uses *Saratoga* to test DTN and Interplanetary Internet on UK-DMC, 2008.

Multiple *Saratoga* implementations in progress with interoperability testing.

Research led to new use

- SSTL remote-sensing images grew to cross 4GiB file size, needing >32-bit pointers.
- How to design a *scalable* file transfer protocol able to handle any size file, without requiring separate incompatible implementations for big files?
- Solved this problem with 16/32/64/128-bit pointers and advertising capabilities.

not needed - yet!

- Support for scalability and streaming introduced new users – high-speed networking for radio astronomy in Very Long Baseline Interferometers.

What is *Saratoga* ?

- Version 0 developed by Surrey Satellite Technology Limited (SSTL) as a replacement to CFDP for simple high speed, low processing, file delivery from Low Earth Orbit to Ground over highly asymmetric links.
- *Saratoga* is a high-speed, UDP-based, peer-to-peer protocol, providing error-free guaranteed delivery of files, or streaming of data.
- Send data packets out as fast as you can. No specified congestion control is required, since data is usually only going one hop over a private link, or across high-speed, low-congestion private networks.
- Some implementations have a rate-limiting option for restricted downstream links where line rate may not match downstream radio link
- No specified timers means no timeouts, so Saratoga is *ideal* for very long propagation delay networks (such as deep space).
- Every so often the transmitter asks for an acknowledgement from the file receiver. The receiver can also send acks if it thinks it needs to, or to start/restart/finish a transfer.
- Acks are Selective Negative Acknowledgements (SNACKs) indicating received packets, and any gaps to fill with resent data, including information so that intelligent sender rate control or congestion control can be provided if needed.
- Any multiplexing of flows is done by the *Saratoga* peers.
- *Saratoga* is an excellent protocol to use in asymmetric network topologies.

Saratoga is a reliable transport over UDP

Simple sliding window with selective acknowledgments.

- The HOLESTOFILL list on the receiver requests the transmitter to re-send frames that have not been properly received (a SNACK) by sending a STATUS with the list of HOLESTOFILL.
- The receive window only advances when offsets are contiguous. The left edge of the transmit window does not advance until the holes have been acknowledged by a HOLESTOFILL frame with an advanced offset.
- The UDP checksum is used per packet to cover both the header and payload. It is consistent, but not that strong (one's complement), and does not provide end-to-end guarantees for payloads sent using multiple packets.
- An optional end-to-end checksum, using one of CRC32/MD5/SHA-1, over the entire file being transferred, increases confidence that a reliable copy has been made, and that fragments have been reassembled correctly.

Optional features of *Saratoga* version 1

Specified to the IETF in an experimental internet-draft. Adds features.

Major features

- Scalable to handle large files. 16-bit descriptors for efficiency with small files. 128-bit descriptors cope with *huge* files up to 2^{128} bytes. 32- and 64-bit descriptors most useful.
- Streaming of data is supported. This allows *Saratoga* to be used for real-time delivery outside the file-based store-and-forward paradigm.

Minor features

- Supports link-local multicast to advertise presence, discover peers and for delivery to multiple receivers simultaneously for e.g. file or code image updates. (Will outperform TFTP trivial file transfer.)
- Optional UDP-Lite use for tolerating errors in payloads and minimizing checksum computation overhead. The UDP-Lite checksum covers a minimum of IP/UDP-Lite/*Saratoga* headers. The header content is always checked so that the information *about* the data is error-free.
- Optional "DTN bundle" delivery as a "bundle convergence layer". Shown with tests from the UK-DMC satellite.

What *Saratoga* does not do

- There is no MTU discovery mechanism, so you have to know the maximum packet size your network can transmit at. i.e. dictated by the frame size. This is okay for your own private network, but would be troublesome if used across the Internet.

- ***Saratoga* does not include "slow-start" or congestion control.** That is considered bad and unsociable behaviour on the Internet. *Saratoga* just blasts away on a link with no regard for other flows - which is the exact behaviour that makes it desirable in private networks and these environments!
  - Simulations have shown that it is possible to implement congestion control mechanisms in *Saratoga* if desired - see our parallel University of Oklahoma paper describing Sender-Based TCP Friendly Rate Control.
    - Saratoga's timestamp option can be used to implement such *closed-loop* mechanisms.
  - Simple *open-loop* rate-limiting output to *XMbps* can also allow Saratoga to coexist with other traffic.

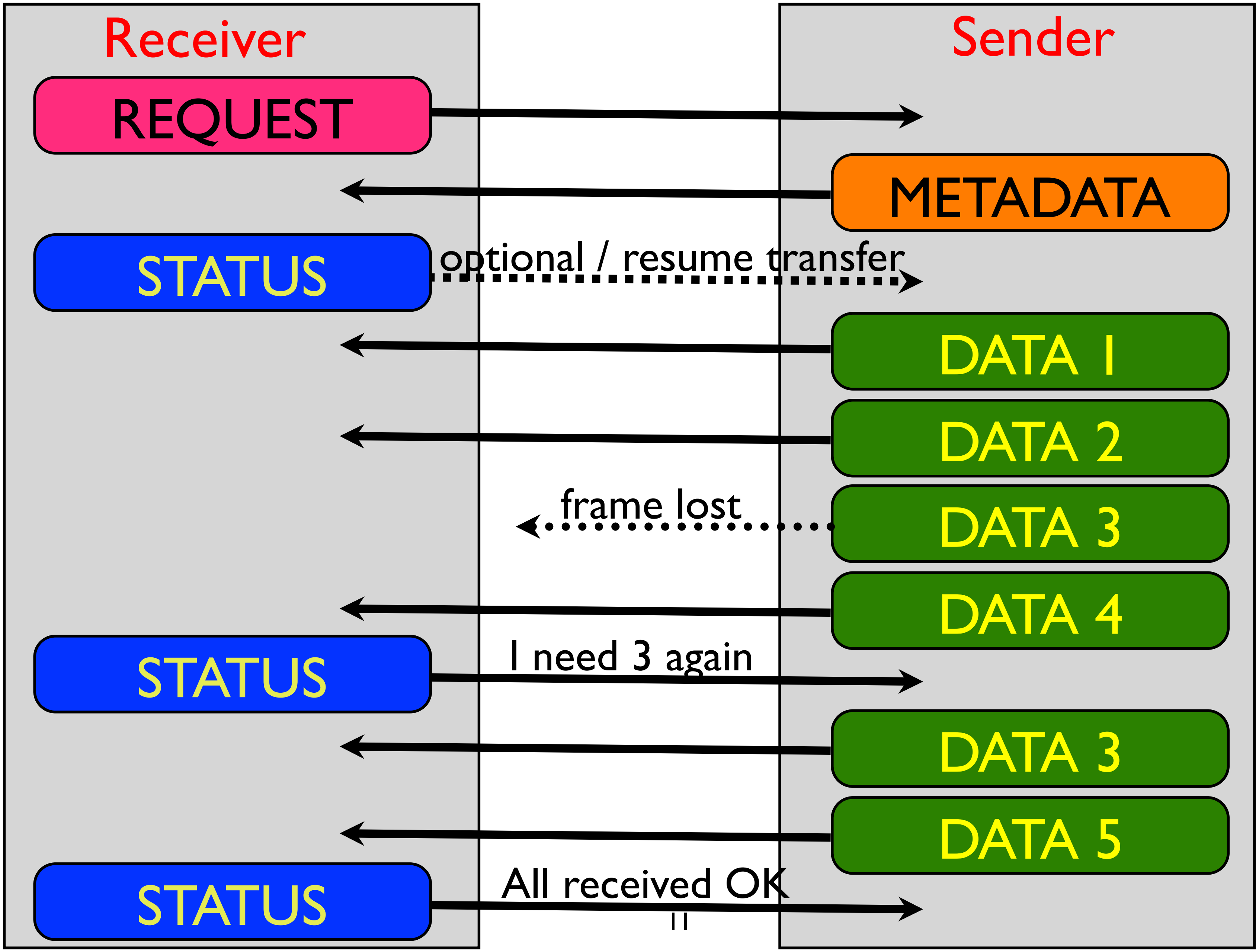
Saratoga Transactions

GET	Get a named file from the peer
GETDIR	Get a directory listing of files from the peer
DELETE	Delete a named file from the peer
PUT	Put a named file or stream data to the peer
PUTDIR	Put a directory listing of local files to the peer

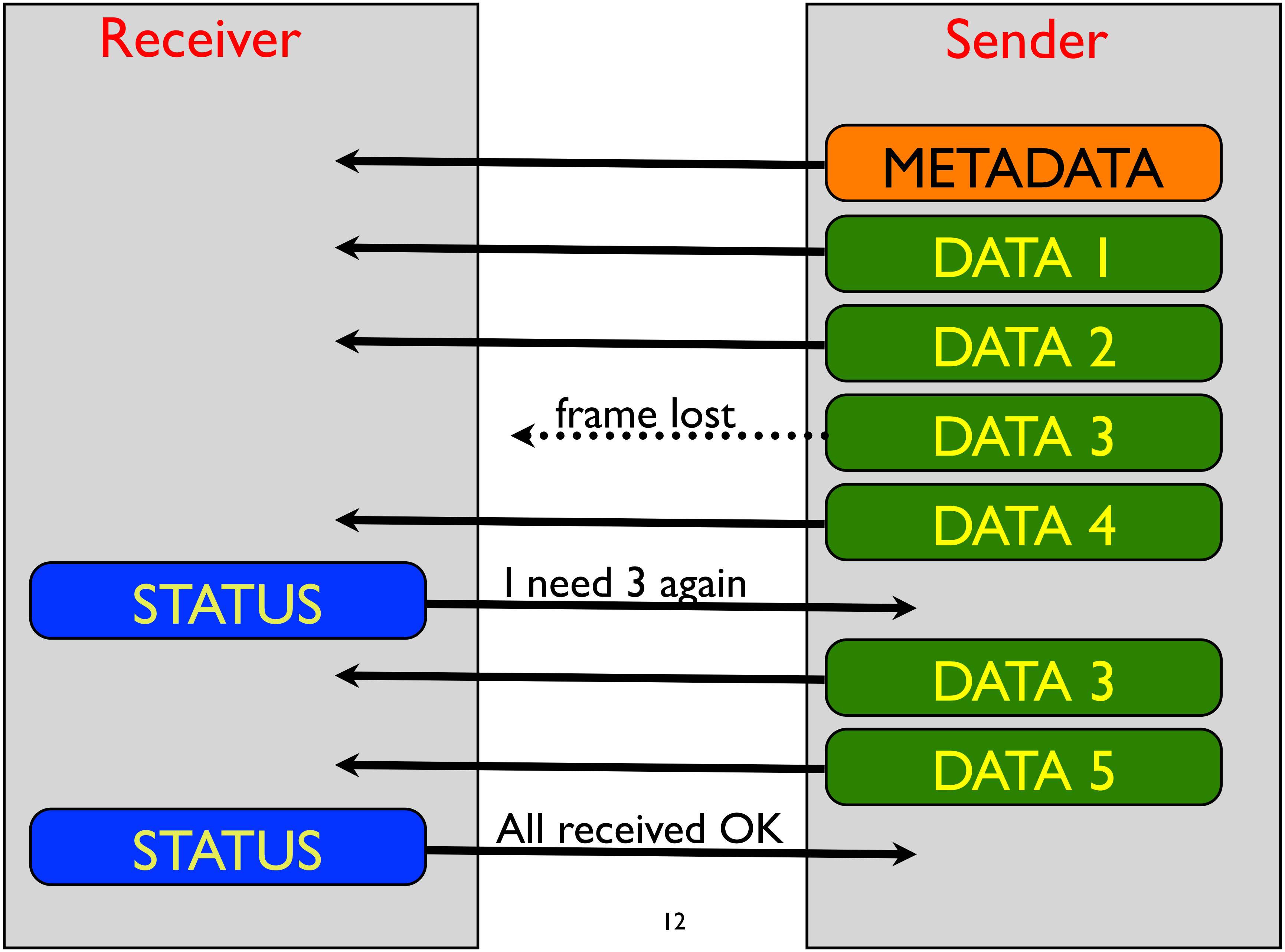
Saratoga Frame Types

BEACON	Sent periodically. Describes the *Saratoga* peer: Identity (e.g. EID) capability/desire to send/receive packets. max. file descriptor handled: 16/32/64/128-bit.
REQUEST	Asks for a file initiating 'get' transaction get file get directory listing delete a file.
METADATA	Sent at start of transaction. Initiates a 'put' transaction. Describes the file, bundle or stream: set identity for transaction file name/details, including size. set descriptor size offsets to be used for this transaction (16/32/64/128-bit pointer sizes.)
DATA	Actual Data. Uses descriptor of chosen size to indicate offset for data segment in the file/bundle or stream. May request an 'ack' (send me a holestofill).
STATUS	Missing Data Offsets / Error & Status Messages Selective negative ack ('snack') HOLESTOFILL data. Set left window edge for successful transfer so far List of offsets and lengths indicate missing 'holes' in data.

Transaction GET or GETDIR



Transaction PUT or PUTDIR



Saratoga Version 1 implementations

PERL (NASA Glenn Research Center)

- Sequential file transfer
- Rate-limiting implemented

C++ (NASA Glenn Research Center)

- Discovery
- Multiplexed file transfer
- Hooks for bundling and streaming
- Rate-limiting to be implemented

C (Charles Smith under contract to Cisco Systems)

- Implementation licensed to CSIRO by Cisco
- Built for Speed (Raw I/O)
- Streaming to be implemented in FPGA
- File transfer may be implemented in FPGA

Wireshark Dissector (Charles Smith)

We hope to make some of these implementations available to the public.

Conclusions

- *Saratoga* is a simple, reliable, transport protocol that can be implemented on low-power low-speed embedded systems, and still give high performance. Should also be implementable in FPGAs and ASICs.

- If you have a high-speed private network and you want to get as much data as possible moved quickly and reliably between peers, then you need a simple, reliable transport protocol.
  - *Saratoga* is a good choice for this application space. (That's why *Saratoga* has been in use since 2004 to download images from SSTL's DMC satellites.)
  - Radio astronomy has high-speed private networks, and needs to move a massive amount of data around. So we're implementing *Saratoga* for radio astronomy.

Development Strategy

- We plan to take Saratoga Version 1 through as an individual submission for Experimental status.
  - Rational: Keep the current implementations moving along and maintain interoperability.
  - Keep detailed discussion on Version 1 to Google Groups saratoga-discussion list
    - Those interested in joining that list, please contact Lloyd Wood (lloyd.wood@gmail.com)

- After working with Version 1 and if there is sufficient interest, then see if TSVWG would be interested in taking this on as a working group document "Saratoga Version 2".

- Comments / Suggestions

Implementations underway

The public *Saratoga* specification has led to:

- **a mature internet-draft**, aiming for IETF RFC.
- **multiple independent implementations** (SSTL, NASA Glenn, CSIRO and Cisco Systems) with interoperability testing underway.
- a simulator showing that TCP friendliness can be supported (University of Oklahoma)

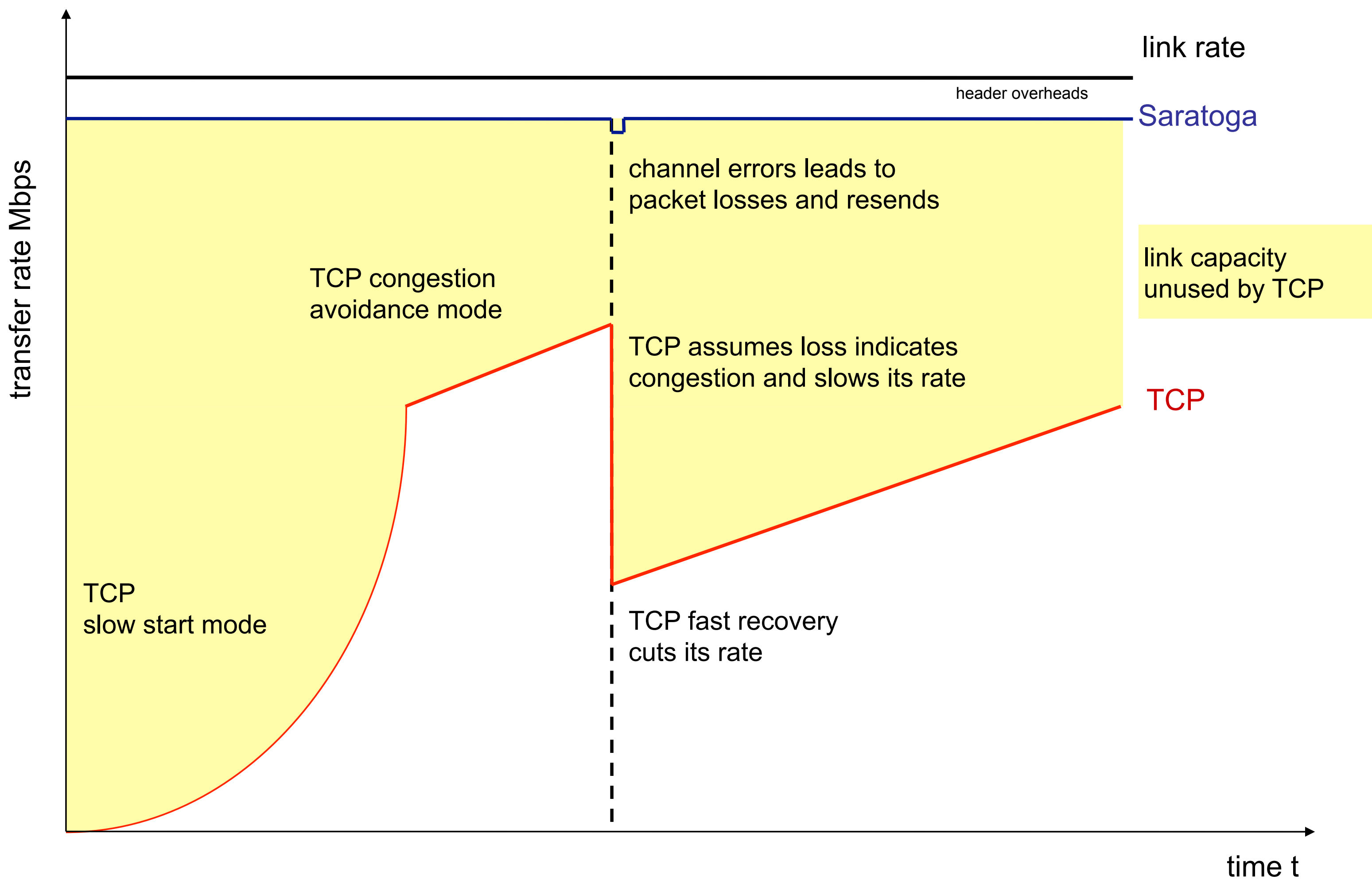
Identified uses for *Saratoga* data delivery:

- **remote-sensing Earth data** from satellites (SSTL) and UAVs (NASA Glenn)
- high-end **radio astronomy sensor data** and processed data cubes (Square Kilometre Array)
- other applications in private networks and in supercomputing.
- could even replace TFTP for fast network booting of Cisco routers and phones...

Why *Saratoga* instead of FTP/TCP ?

- For high throughput and link utilization on dedicated links, where a single TCP flow cannot fill the link to capacity.
- For links where TCP's assumptions about loss/congestion/ competition simply don't hold. i.e. High speed bulk transfer.
- There is no such thing as "slow-start" specified in *Saratoga*.
- Able to cope with high forward/back network asymmetry (>850:1).
- Long path-delay use – eventually TCP will fail to open a connection because its SYN/ACK exchange won't complete. TCP has many unwanted timers.
- Simplicity. TCP is really for a conversation between two hosts; needs a lot of code on top to make it transfer files. A focus on just moving files or streams of data makes sequence numbering simpler.
- Having SNACKs means that handling a sequence number wraparound when in streaming or bundling mode becomes easy.

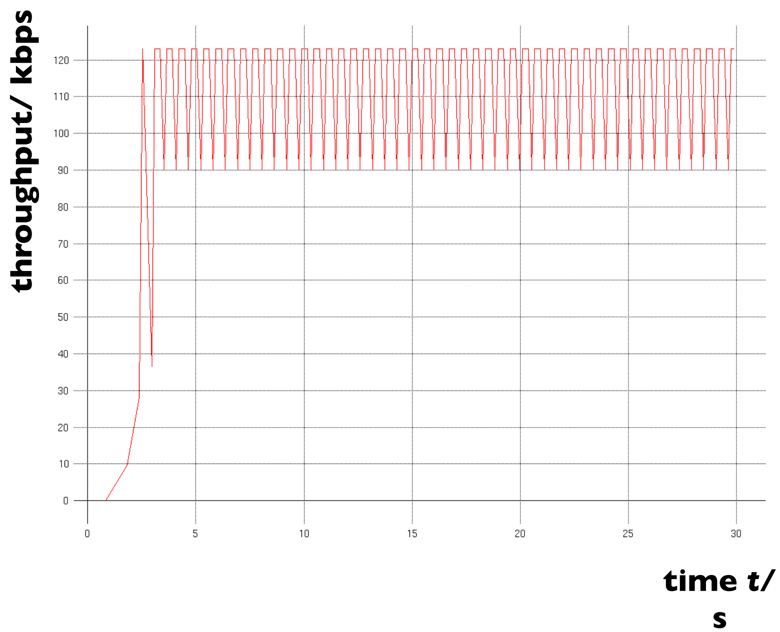
Why *Saratoga* instead of FTP/TCP ?

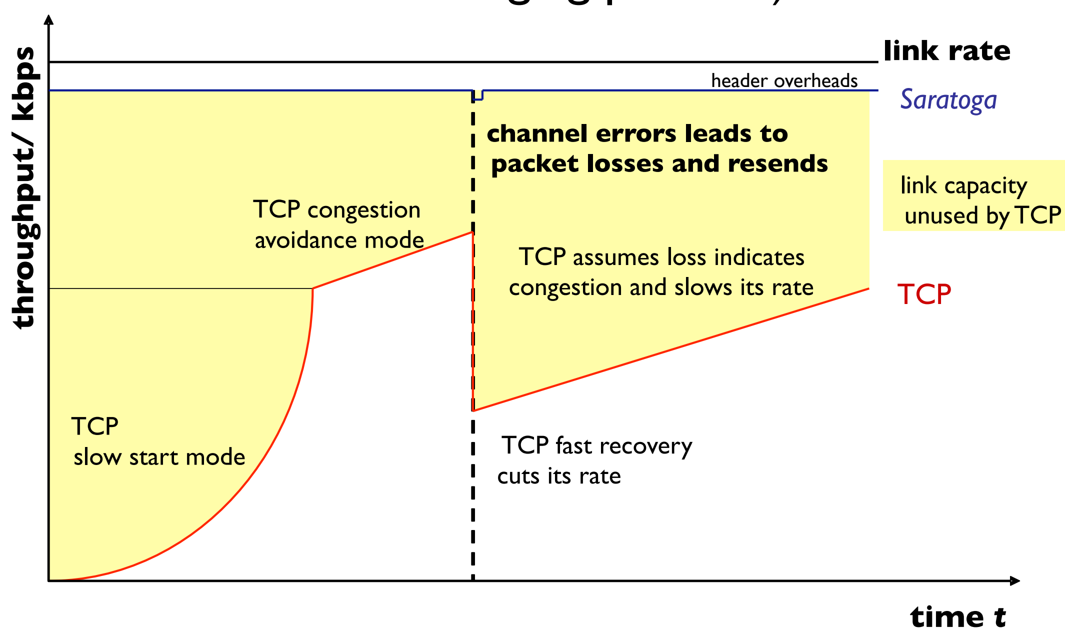


Saratoga's approach

Run as fast as possible, at maximum possible rate over a private dedicated link. Deliberately **don't** emulate TCP's cautious congestion-control behaviour.

('TCP friendly' behaviour can be added without changing packets.)





A single TCP flow can't fill a link – reaches capacity, then backs off.

A single *Saratoga* flow can take advantage of all the available capacity.